Acadgild

Full-stack Web Development

Acadgild

Full-stack
Web Development

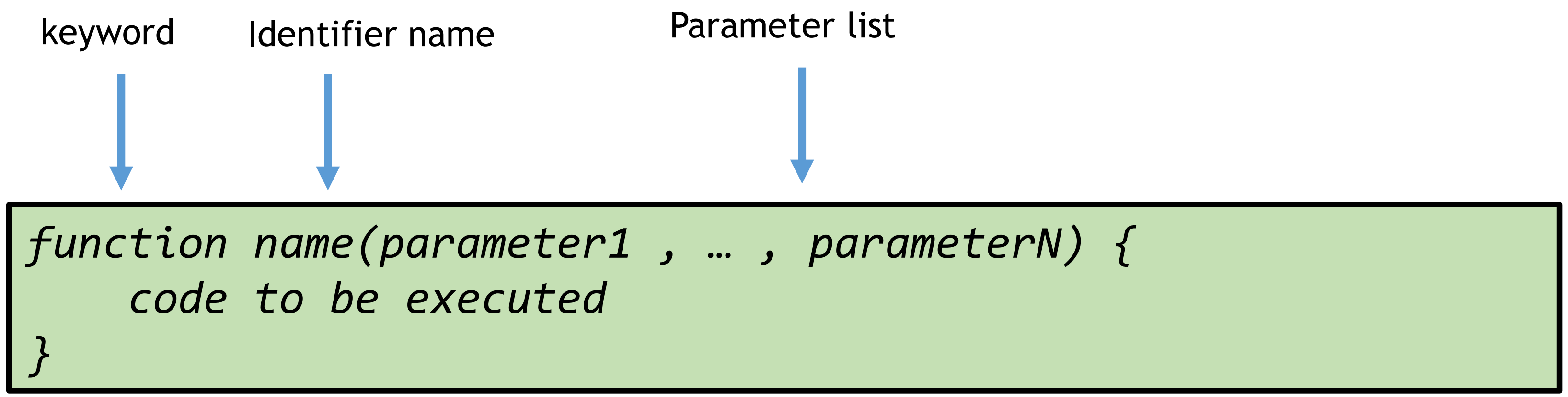
Agenda

What is a Function?

- One of the fundamental building blocks in JavaScript
- A set of statements or block of code that performs a task or calculates a value
- Can be called anywhere
- Should be defined before it is called

Definition of Function

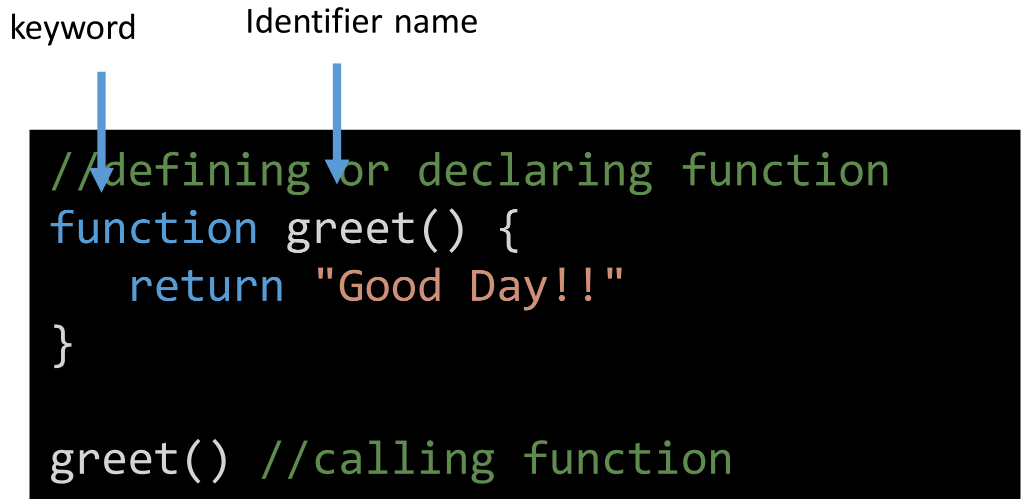
- Function definition has following parts:
  - **function** keyword
  - followed by a unique function **name**
  - a list of parameters wrapped in a pair of **parentheses ()**
  - a block of code surrounded by curly brackets **{}**
- Function names follow the same naming rules as variables



```
function name(parameter1 , … , parameterN) {
    code to be executed
}
```

Calling Function

- Defining a function does not execute it
- Defining the function simply names the function and specifies what to do when the function is called
- Calling the function actually performs the specified action
- Call or Invoke a function with its name somewhere later in the code



```
//defining or declaring function
function greet() {
   return "Good Day!!"
}

greet() //calling function
```

Function Parameters

- Are inputs that get passed into functions
- Listed inside the parentheses () in the function definition
- Inside the function, parameters behave as local variables
- Can take multiple parameters separated by comma
- Also called arguments

Parameter list

```
function wish(firstName, lastName) {
    var fullName = firstName + " , " + lastName;
    return fullName;
}
```

Default Parameters

- In JavaScript, parameters of functions default to undefined
- With default parameters you can set a different default value
- Default parameters are introduced in ES2015
- Must be rightmost in parameter list

Default parameter discount will receive value 10 when not supplied at function call

```
calculatePrice(200, 8); /* discount parameter not
passed , hence default value will be 10 instead of
undefined */
```

Return Statement

- Function can have an optional **return** statement
- Required if you want to return a value from a function
- Return value is "returned" back to the "caller"
- With return statement function execution will be completed

Return statement

```
function wish(firstName, lastName) {
    var fullName = firstName + " , " + lastName;
    return fullName;
}
```

Anonymous Functions

- Function created without a name
- Cannot be called later if not assigned or referenced to a variable

```
function () {
        alert("clicked")
}
```

Function Expressions

- Functions can be assigned to variables. Such assignments are known as function expressions

Anonymous Function

```
var handleClick = function () {
      alert("clicked")
}
```

A variable assigned with function

Callback Functions

- A function passed as a parameter to another function and executed by another function

```
<script>
var button = document.getElementById("btnClickMe");

function handleClickMe() {
        alert("button clicked!!!")
}
button.addEventListener('click', handleClickMe);
</script>
```

callback

Scope

Scope determines the accessibility (visibility) of variables.
In JavaScript there are two types of scope:

- Local Scope
- Global Scope

Local scope

- Every function has its own local scope
- A variable that is declared inside a function definition is local. It is created and destroyed every time the function is executed, and it cannot be accessed by any code outside the function

Global scope

- There is only one Global scope throughout a JavaScript
- A variable that is declared outside a function definition is a global variable, and its value is accessible and modifiable throughout your program

Scope – Example

```
<script>
//global scope
var pName = 'iphone';

function upper(text) {
    //local scope
    var result = text.toUpperCase();
    console.log(result);
    console.log(pName); // global variable pName accessible
    inside function
}

console.log(result); //error, local variable result not
accessible outside function
</script>
```

Email us - support@acadgild.com